

mod_rewrite

Introduction to mod_rewrite

Rich Bowen, Web Guy, Asbury College

rbowen@apache.org

aka DrBacchus

<http://people.apache.org/~rbowen/>

ApacheCon Europe, 2006



Outline

- Regex basics
- RewriteRule
- RewriteCond
- RewriteMap
- The evils of .htaccess files
- Assorted odds and ends



mod_rewrite is not magic

- Fear, more than complexity, makes mod_rewrite difficult





Although, it is complex

“The great thing about
mod_rewrite
is it gives you all the configurability
and flexibility of Sendmail. The downside to
mod_rewrite is that it gives you all
the configurability and flexibility of
Sendmail.”

-- Brian Behlendorf

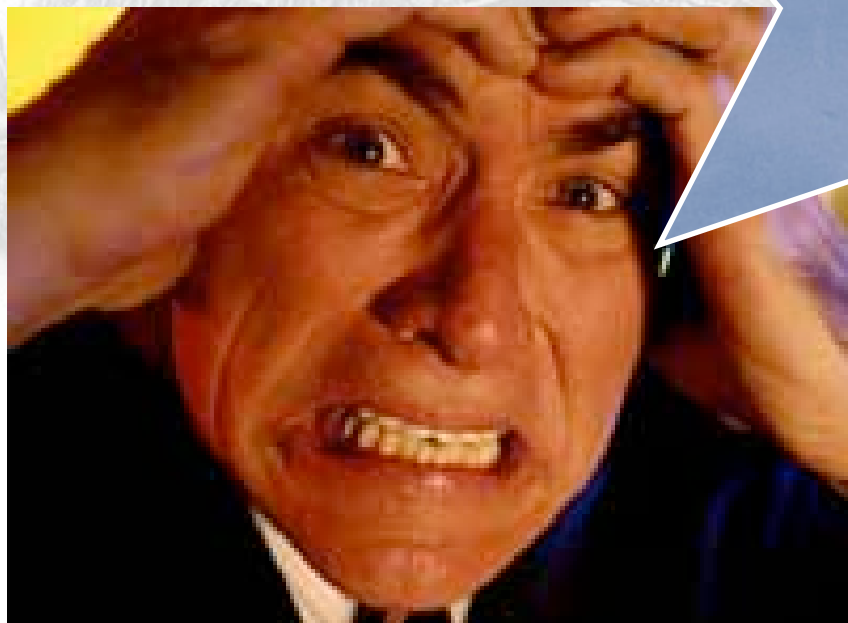
And let's not forget voodoo!

`` Despite the tons of
examples and docs,
mod_rewrite is voodoo.
Damned cool voodoo,
but still voodoo. "
-- Brian Moore



Line noise

"Regular expressions
are just line noise.
I hate them!"
(Heard 20 times per
day on IRC)



When you hear it often
enough, you start to
believe it



Now that that's out of the way

- Regular expressions are not magic
- They are an algebraic expression of text patterns
- Once you get over the mysticism, it can still be hard, but it's no longer mysterious

$$y_2 - y_1 = mx_2 + b - (mx_1 + b)$$

$$y_2 - y_1 = mx_2 + b - mx_1 - b$$

$$y_2 - y_1 = mx_2 - mx_1$$

$$y_2 - y_1 = m(x_2 - x_1)$$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$



Vocabulary

- We're going to start with a very small vocabulary, and work up from there
- Most of the time, this vocabulary is all that you'll need



•

- . matches any character
- "a.b" matches acb, axb, a@b, and so on
- It also matches Decalb and Marbelized



+

- + means that something needs to appear one or more times
- "a+" matches a, aa, aaa, and Stellaaaaaaaaaaaaa!
- The thing that is repeating isn't necessarily just a single character



*

- * means that the previous thingy needs to match zero or more times
- This is subtly different from + and some folks miss the distinction
- "giraf*e" matches giraffe and girafe
- It also matches girae



?

- ? means that the previous thingy needs to match zero or one times
- In other words, it makes it optional
- “colou?r” matches color and colour



^

- ^ is called an anchor
- It requires that the string start with the pattern
- ^A matches ANDY but it does not match CANDY
- Pronounced “hat” or “caret” or “circumflex” or “pointy-up thingy”



\$

- \$ is the other anchor
- It requires that the string end with the pattern
- a\$ matches canada but not afghanistan



()

- () allows you to group several characters into one thingy
- This allows you to apply repetition characters (*, +, and ?) to a larger group of characters.
- "(ab)+" matches ababababababab



(), continued

- () allows you to capture a match so that you can use it later.
- The value of the matched bit is stored in a variable called a backreference
- It might be called \$1 or %1 depending on the context
- The second match is called \$2 (or %2) and so on



[]

- [] defines a “character class”
- [abc] matches a or b or c
- “c[uoa]t” matches cut, cot, or cat
- It also matches cote
- It does not match coat



NOT

- In `mod_rewrite` regular expressions, `!` negates any match
- In a character class, `^` negates the character class
- `[^ab]` matches any character except for `a` or `b`.



So, what does this have to do with Apache?

- mod_rewrite lets you match URLs (or other things) and transform the target of the URL based on that match.

```
RewriteEngine On  
# Burninate ColdFusion!  
RewriteRule (.*)\.cfm$ $1.php [PT]  
# And there was much rejoicing. Yaaaay.
```

RewriteEngine

- “RewriteEngine On” enables the mod_rewrite rewriting engine
- No rewrite rules will be performed unless this is enabled in the active scope
- It never hurts to say it again





RewriteLog

```
RewriteLog /www/logs/rewrite_log  
RewriteLogLevel 9
```

You should turn on the RewriteLog before you do any troubleshooting.



RewriteRule

RewriteRule pattern target [flags]

- The pattern part is the regular expression that you want to look for in the URL.
- If they try to go HERE send them HERE instead.
- The behavior can be further modified by the use of one or more flags



Example 1

- SEO – “Search Engine Optimization”
- Also known as “convenient way to siphon money off of people who don’t understand search engines.”
- Typical strategy is to make “clean URLs”



URL beautification

- A URL looks like:

<http://example.com/cgi-bin/book.cgi?author=bowen&topic=apache>

We would prefer that it looked like

<http://example.com/book/bowen/apache>

It's easier to type, and easier to remember



Example 1, cont'd

```
RewriteRule ^/book/(.*)/(.*) \  
/cgi-bin/book.cgi?topic=$1&author=$2 [PT]
```

- User does not notice that the transformation has been made
- Used \$1 and \$2 to capture what was requested
- Slight oversimplification. Should probably use $([^\wedge/]+)$ instead.



Flags

- Flags can modify the behavior of a RewriteRule
- I used a flag in the example, and didn't tell you what it meant
- So, here are the flags

France



Italy



Belgium



Great Britain



Roumania



Serbia



Russia



Greece



China



Portugal



Guatemala



Brazil



Siam



Liberia

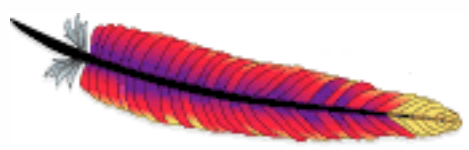


Cuba



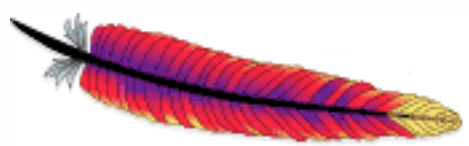
Nicaragua





By the way ...

- Default is to treat the rewrite target as a file path
- If the target starts in `http://` or `https://` then it is treated as a URL, and a [R] is assumed (Redirect)
- In a `.htaccess` file, or in `<Directory>` scope, the file path is assumed to be relative to that scope



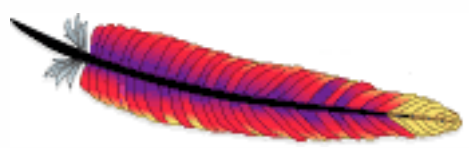
RewriteRule flags

- [Flag] appears at end of RewriteRule
- More than one flag separated by commas
- I recommend using flags even when the default is what you want – it makes it easier to read later
- Each flag has a longer form, which you can use for greater readability.
- There's *lots* of flags



Chain

- [C] or [Chain]
- Rules are considered as a whole. If one fails, the entire chain is abandoned

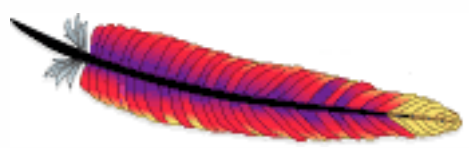


Cookie

- [CO=NAME:Value:Domain[:lifetime[:path]]]
- Long form [cookie=...]
- Sets a cookie

```
RewriteRule ^/index.html - [CO=frontdoor:yes:.example.com]
```

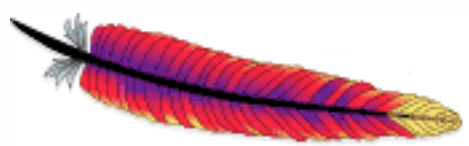
In this case, the default values for path ("/") and lifetime ("session") are assumed.



Env

- [E=var:val]
- Long form [env=...]
- Sets environment variable
- Note that most of the time, SetEnvIf works just fine

```
RewriteRule \.jpg$ - [env=dontlog:1]
```

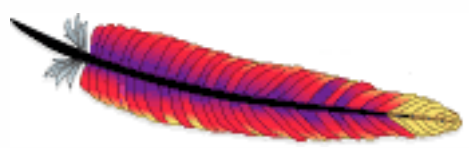
Forbidden

- [F] or [Forbidden] forces a 403 Forbidden response
- Consider mod_security instead for pattern-based URL blocking

```
RewriteEngine On  
RewriteRule (cmd|root)\.exe - [F]
```

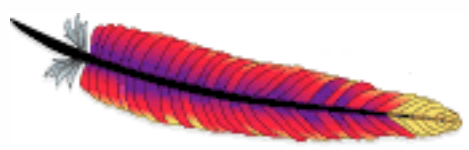
You could use this in conjunction with [E] to avoid logging that stuff

```
RewriteRule (cmd|root)\.exe - [F,E=dontlog:1]  
CustomLog /var/log/apache/access_log combined \  
env=!dontlog
```



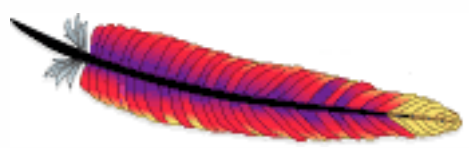
Handler

- [H=application/x-httpd-php]
- Forces the use of a particular handler to handle the resulting URL
- Can often be replaced with using [PT] but is presumably slightly faster
- Available in Apache 2.2



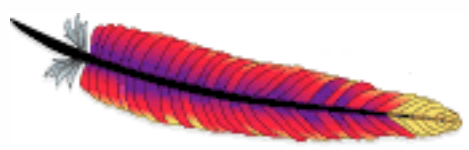
Last

- [L] indicates that you've reached the end of the current ruleset
- Any rules following this will be considered as a completely new ruleset
- It's a good idea to use it, even when it would otherwise be default behavior. It helps make rulesets more readable.



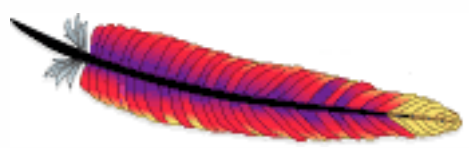
Next

- The [N] or [Next] flag is a good way to get yourself into an infinite loop
- It tells mod_rewrite to run the entire ruleset again from the beginning
- Can be useful for doing “global search and replace” stuff
- I find RewriteMap much more useful in those situations



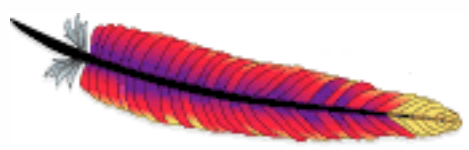
NoCase

- [NC] or [nocase] makes the RewriteRule case insensitive
- Regexen are case-sensitive by default



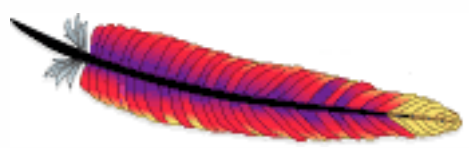
NoEscape

- [NE] or [noescape] disables the default behavior of escaping (url-encoding) special characters like #, ?, and so on
- Useful for redirecting to a page #anchor



NoSubreq

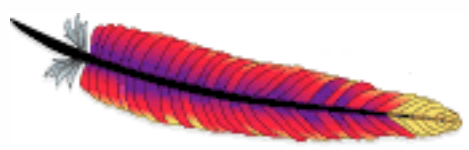
- [NS] or [nosubreq] ensures that the rule won't run on subrequests
- Subrequests are things like SSI evaluations
- Image and css requests are NOT subrequests



Proxy

- [P] rules are served through a proxy subrequest
- mod_proxy must be installed for this flag to work

```
RewriteEngine On  
RewriteRule (.*)\.(jpg|gif|png) \  
http://images.example.com$1.$2 [P]
```

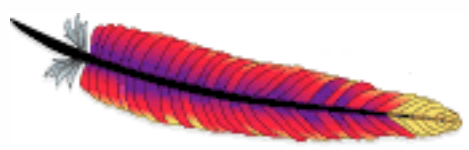
Passthrough

- [PT] or [passthrough]
- Hands it back to the URL mapping phase
- Treat this as though this was the original request



QSAppend

- [QSA] or [qsappend] appends to the query string, rather than replacing it.



Redirect

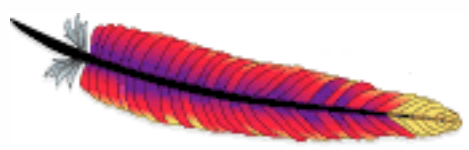
- [R] or [redirect] forces a 302 Redirect
- Note that in this case, the user will see the new URL in their browser
- This is the default behavior when the target starts with http:// or https://



Skip

- [S=n] or [skip=n] skips the next n RewriteRules
- This is very weird
- I've never used this in the real world. Could be used as a sort of inverse RewriteCond (viz WordPress)

RewriteRule %{REQUEST_FILENAME} -f [S=15]



Type

- [T=text/html]
- Forces the Mime type on the resulting URL
- Used to do this instead of [H] in some contexts
- Good to ensure that file-path redirects are handled correctly

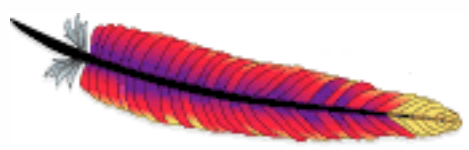
```
RewriteRule ^(.+\.php)s$ $1 [T=application/x-httpd-php-source]
```



RewriteCond

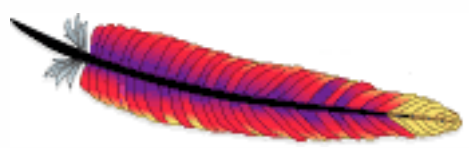
- Causes a rewrite to be conditional
- Can check the value of any variable and make the rewrite conditional on that.

```
RewriteCond TestString Pattern [Flags]
```

RewriteCond

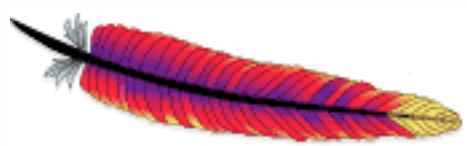
- The test string can be just about anything
- Env vars, headers, or a literal string
- Backreferences become %1, %2, etc



Looping

- Looping occurs when the target of a rewrite rule matches the pattern
- This results in an infinite loop of rewrites

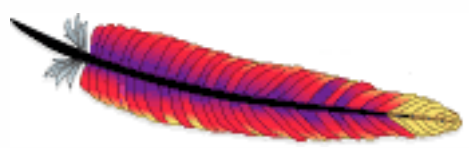
```
RewriteCond %{REQUEST_URI} \  
    !^/example.html  
RewriteRule ^/example /example.html [PT]
```

Conditional rewrites

- Rewrites conditional on some arbitrary thingy
- Only first Rule is dependent

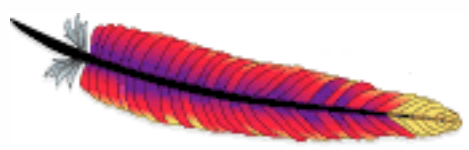
```
RewriteEngine on
RewriteCond %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule ^page\.html$ page.day.html
RewriteRule ^page\.html$ page.night.html
```



SSL Rewrites

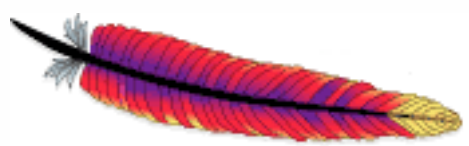
- Redirect requests to http:// if they are for the “wrong” hostname

```
RewriteCond %{HTTP_HOST} !^www.righthost.com$  
RewriteRule (.*?) http://%{HTTP_HOST}$1 [R]
```

RewriteMap

- Call an external program, or map file, to perform the rewrite
- Useful for very complex rewrites, or perhaps ones that rely on something outside of Apache



RewriteMap - file

- File of one-to-one relationships

```
RewriteMap docsmap txt:/www/conf/docsmap.txt  
RewriteRule ^/docs/(.*) ${docsmap:$1} [R,NE]
```

Where docsmap.txt contains:

```
Alias http://httpd.apache.org/docs/mod\_alias.html#alias  
Redirect http://httpd.apache.org/docs/mod\_alias.html#redirect  
... etc
```

Requests for <http://example.com/docs/something> now get redirected to the Apache docs site for 'something'. [NE] makes the #anchor bit work.



Poor-man's load balancing

- Random selection of server for “load balancing”

```
RewriteMap servers rnd:/www/conf/servers.txt  
RewriteRule (.*) http://${servers:loadbalance}$1 [P,NS]
```

servers.txt contains:

```
loadbalance mars|jupiter|saturn|neptune
```

Requests are now randomly distributed between the four servers. The 'NS' ensures that the proxied URL doesn't get re-written.



RewriteMap – program

- Call an external program to do the rewrite
- Perl is a common choice here, due to its skill at handling text.

```
RewriteMap dash2score \  
    prg:/usr/local/apache/conf/dash2score.pl  
RewriteEngine On  
RewriteRule (*.*) ${dash2score:$1} [PT]
```




dash2score.pl

```
#!/usr/bin/perl
$| = 1; # Turn off buffering
while (<STDIN>) {
    s/-/_/g; # Replace - with _ globally
    print $_;
}
```

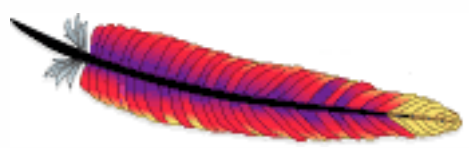
Turning off buffering is necessary because we need the output immediately for each line we feed it.

Apache starts the script on server startup, and keeps it running for the life of the server process



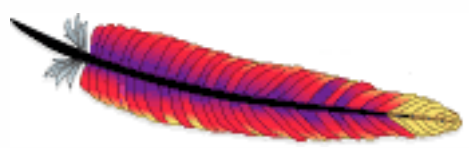
.htaccess files

- .htaccess files are evil
- However, a lot of people have no choice
- So ...



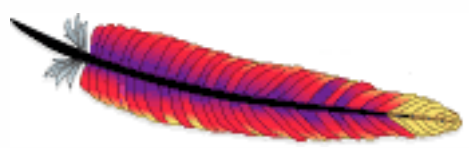
.htaccess files

- In .htaccess files, or <Directory> scope, everything is assumed to be relative to that current scope
- So, that scope is removed from the RewriteRule
- `^/index.html` in `httpd.conf` becomes `^index.html` in a .htaccess file or <Directory> scope



.htaccess files

- RewriteLog is particularly useful when trying to get .htaccess file RewriteRules working.
- However, you can't turn on RewriteLog in a .htaccess file, and presumably you're using .htaccess files because you don't have access to the main server config.
- It's a good idea to set up a test server on your home PC and test there with RewriteLog enabled



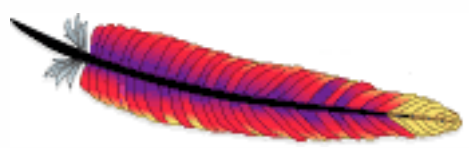
.htaccess files

- The rewrite pattern is relative to the current directory
- The rewrite target is also relative to the current directory
- In httpd.conf, the rewrite target is assumed to be a file path. In .htaccess files, that file path is relative to the current directory, so it seems to be a URI redirect.



Further resources

- <http://rewrite.drbacchus.com/>
- “Definitive Guide to mod_rewrite” by Rich Bowen, from APress
- <http://httpd.apache.org/docs-2.2/rewrite/>



Questions?

